Additional project Report
(New algorithm vs Dijkstra)

Aim:

The aim of the project is to develop an algorithm to find the shortest distance between two nodes that works better than the Dijkstra algorithm.
The nodes here represent Indian states(20 states are used).

Data Format:

The distance between all the neighbouring states was collected. They were stored in a weight adjacency matrix.

Overview of the algorithm:

We used the concept of k-trees. A k-tree is a graph that can be reduced to the k-complete graph by sequentially removing k-degree vertices with completely connected neighbours.
We assumed that the graph follows 3-trees as most of the states has three neighbouring states.

The algorithm involves two steps:
1. bottom to top
2. top to bottom

Step 1: bottom to top
- We remove nodes from the graph till we remain with four nodes(one of the four nodes contain the source node from where we want to find the distance).
- While removing each node, we store the neighbours that are present currently at the time of removing that node. This information will be is used in the "top to bottom" step. Also, we update the weights and degree of its neighbours.
- We will be left with 3 nodes along with the source node.
- We find the distance between the source node and the other three nodes, which is the shortest distance from the source node.
(We store the nodes in a stack after removing it from the graph).

Step 2: bottom to top
- We add nodes one by one to the graph(we remained in the "top to bottom" step) by removing the nodes from the stack.
- We find the shortest distance from source using it's updated neighbour's shortest distance.

Now, we can find the shortest distance to any node from the source node.

Results:

We compared runtime between our algorithm and the Dijkstra algorithm. The results are shown below:

	New algorithm	Dijkstra algorithm
Case1:	0.0210694	0.029745
Case2:	0.0247875	0.024788
Case3:	0.029745	0.032224
Case4:	0.029745	0.027266
Case5:	0.01983	0.029745
Case6:	0.0322238	0.049575
Case7:	0.0446175	0.044618
Case8:	0.0371813	0.039660
Case9:	0.0272663	0.039660
Case10:	0.0272663	0.047096

The new algorithm, outperformed in most of the cases when compared to the Dijkstra algorithm in terms of runtime.